LEC 07

CSE 121

Methods, Parameters, Returns

BEFORE WE START

Talk to your neighbors:

Is cereal a soup?

Music: CSE 121 25au Lecture Tunes

Instructors: Brett Wortzman & James Weichert

TAs:

Trey	Ava	Caleb	Elden	Anya
Amogh	Reese	Anum	Suyash	Minh
Samrutha	Hayden	Abdul	Sthiti	TJ
Dalton	Aki	Janvi	Paul	Zach
Ailsa	Spencer	Navya	Shayna	Cayden
Ryan	Savannah	Sam	Jesse	Johnathan
Anant	Tamsyn	Jessica	Nhan	

Questions during Class?

Raise hand or send here

sli.do #cse121

Agenda

- **Announcements, Reminders**
- P1 Preview
- Revisiting Class Constants
- More Methods!

Announcements, Reminders

- **P1** is out, due next Tuesday, October 21st
  - start early – this one is tough!
- **R1** released yesterday, due Thursday October 23rd
- **Quiz 0** was yesterday
  - grades will be out before Quiz 1

C1 ASCII Art Creations: Cats (+ Friends)

```
Task 1:A happy alpaca
           |\   /|
           | · x · |
\________/        |
|                 |
 \_____________/
   /        /
```

```
Task 2: Three cats holding a star
 /\___/\
(=0 . 0=)
/  >*<  \

 /\___/\
(=0 . 0=)
/  >*<  \

 /\___/\
(=0 . 0=)
/  >*<  \
```

```
Task 1: A cat!
__^____^__

| o    o |
|=   w  =|
__________
```

```
() ()
(oto)
(u u)@
```

```
Task 1: Curtis the Cautious Cat!
 ^______^
 / '    ' \
>\    '    /<
  |_  ^  _|
```

```
Task 1: a grumpy cat!
^   ^
 . .
_ ^ _
 ___
```

```
Creative Option 2: Tummy Troubles
As mr. monkey eats bananas, his stomach starts to stretch...
The lengths are often random so he's always full of dread :(
 / \
c ..
(( o)<_)
----- ||
/   \//
|    |
|    |
|    |
|    |
|    |
|    |
|    |
monkey ate 7 bananas
```

```
Task 1: Betty the bunny
/)/)
(..)
(  )
 UU
```

```
Task 1: a bunny!
()()
(- -)
o(0)
```

```
Task 3: 3 cats!!!
  /\___/\        /\___/\        /\___/\
 =       =      =       =      =       =
*|  0   0  |* *|  0   0  |* *|  0   0  |*
*=    V    =* *=    V    =* *=    V    =*
 *|  -^-  |*   *|  -^-  |*   *|  -^-  |*
   \     /       \     /       \     /
```

```
Task 1: A very round cat!
  _/\__/\_
 / -0-0-   \
|  --   --  |
\ v------vv /
```

```
Task 1: An owl
(\_._/).
 (o>o) .
(|^^^|).
--m-m--.
```

```
Task 1: A staring cat.
 ^ _ ^
= 0+0 =
 ==o==
|\   /|
 -u-u- ___//]
```

```
Task 1: A Cutesy piggie
   _____
 ^..^    \9
 (oo)_____/
   WW   WW
```

```
Task 3: The rabbit-like creature family photo
 n n      n n      n n      n n
 (^^)     (^^)     (^^)     (^^)
  ()       ()       ()       ()
 *  *     *  *     *  *     *  *
```

```
Task 1: A liitle mouse
/\__/\
(. . )      ~
|-.-- \   /
|^  ^  \/
----------
```

```
Task 1: A bunny!
 () ~ ()
( ' * ' )
  ~~~~~
( U   U )
```

```
 .^.
 -|-
 -\-  /
```

```
Creative Option 1: cat phalanx
  *     *      *     *      *     *      *     *
 * *   * *    * *   * *    * *   * *    * *   * *
*   ***   *  *   ***   *  *   ***   *  *   ***   *
*  @   @  *  *  @   @  *  *  @   @  *  *  @   @  *
*    ^    *  *    ^    *  *    ^    *  *    ^    *
* * * * * *  * * * * * *  * * * * * *  * * * * * *

  *     *      *     *      *     *      *     *
 * *   * *    * *   * *    * *   * *    * *   * *
*   ***   *  *   ***   *  *   ***   *  *   ***   *
*  @   @  *  *  @   @  *  *  @   @  *  *  @   @  *
*    ^    *  *    ^    *  *    ^    *  *    ^    *
* * * * * *  * * * * * *  * * * * * *  * * * * * *
```

```
                _
               /.>
              / /
             / /
            / /
           / /
          / /
         / /
        / /
       / /
      / /
     / /
    / /
 _________/ /
<__________/
           \_
```

```
Task 3: 3 ASCII arts of a small simple creatures with round triangular bodies and
pointy triangular ears, holding a flower stem, repeated horizontally)
✿/\_/\    ✿/\_/\    ✿/\_/\
 /\◉.◉ \   /\◉.◉ \   /\◉.◉ \
( ɔ\c   ) ( ɔ\c   ) ( ɔ\c   )
-------   -------   -------
```

```
Creative Option 1: a cluster of 9 frogs
 @..@     @..@     @..@
(----)   (----)   (----)
( >__< ) ( >__< ) ( >__< )

 @..@     @..@     @..@
(----)   (----)   (----)
( >__< ) ( >__< ) ( >__< )

 @..@     @..@     @..@
(----)   (----)   (----)
( >__< ) ( >__< ) ( >__< )
```

```
Creative Option 2: a small simple creature with a round triangular body
triangular ears, that is a random number (17) characters wide, holding
✿/\_________/\
 /\◉_________◉ \
( ɔ\c            )
----------------
```

```
Task 3: Birds on the ground
   _         _         _         _
  /.>       /.>       /.>       /.>
____/ /   ____/ /   ____/ /   ____/ /
<_____/   <_____/   <_____/   <_____/
  \_        \_        \_        \_
```

```
Task 3: A line of 3 birds!
  ___        ___        ___
 (o o)      (o o)      (o o)
 (  v  )    (  v  )    (  v  )
//-m--m--  //-m--m--  //-m--m--
```

```
Task 1: A penguin
  _ _ _
 /     \
 | '  ' |
 \___v___/
 |_|_|_|_|_|
 / / |__| \
 \_/ |__|\_/
  \_______/
   vv    vv
```

```
 \_______/
 /•       •\
/   _____   \
|  /     \   |
\  \ _|_ /   /
 |   / \    |
```

```
Task 3: FIVE HIPPOS WHAT IS GOING ON
 @_____@    @_____@    @_____@    @_____@    @_____@
(0      0) (0      0) (0      0) (0      0) (0      0)
(   O O  ) (   O O  ) (   O O  ) (   O O  ) (   O O  )
---------  ---------  ---------  ---------  ---------
```

C1 ASCII Art Creations: Robots (+ Friends)

Task 1: ASCII Robot

```
 |_|
[o o]
 :.:
{ooo}
 / \
```

Task 1: A hopefully friendly robot?

```
 _|__|_
|o  o|
|_==_|
```

Task 3: Five horizontal cakes

Creative Option 2: Chef with random size hat!

```
  _____
 (     )
  |   |
  |   |
  |   |
  |___|
    O
   /|\
    |
   / \
```

Task 1: Man with very big eyes

```
#####
[0 0]
  -
 -|-
  |
  /\
```

Task 1: dragon ASCII art

```
 __)\ (\__/) /(__
__)_ \(6`'6)/ _(__
 )__   \oo/   __(
  )_ `/(  )\ '_(
    \(_>UU<_)/
```

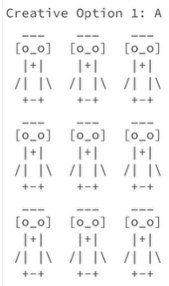
Task 1: a snowman!

```
      (   )
     (.   .)
  |   (   ><  )
 ---(         )---
   ((         ))  |
   ((         ))
     ----------
```

Task 1: Calcifer the Flame

```
          /\
         /  \
        /    \
   /\  /      \
  /  \/        \
  |  (0)    (0) |
   \  -___-___-  /
```

Task 1: A collection of robots



Task 1: I create a little elf with eyes mouth and blush

```
---^-----^--
| / ì   í \ |
|/@    ^   @\|
----->0<----
```

Task 1: a lovely framed photo of a mountain range.

```
o-----o
|  o  |
|^v^v^|
o-----o
```

Task 1: This is a drawing of dancing squidward

```
    ⌒  ⌒
   ( 0  0 )
    --J--
ᵛ^ᵛ^ᵛ^||ᵛ^ᵛ^ᵛ^
   __|_)__
   \     /
  --     --
```

Task 1: Plankton from Spongebob

```
  o   o
   \_/
\ ( 0 ) /
 \|   |/
  (   )
   ---
   | |
```

Creative Option 2: Man with random length neck

```
  _
 /.\
  -
  |
  |
  |
  |
  |
--|--
  |
  -
 / \
```

```
 ^^^^^
<     >
 -----
  ! !

 ^^^^^
<     >
 -----
  ! !
```

Task 3: A line of 3 cups of coffee

```
 ( (      ( (      ( (
  ) )      ) )      ) )
  ....     ....     ....
 |    |]  |    |]  |    |]
 \____/   \____/   \____/
```

Up Next: P1 Preview

- Announcements, Reminders
- **P1 Preview**
- Revisiting Class Constants
- More Methods!

Up Next: Class Constants

- Announcements, Reminders
- P1 Preview
- **Revisiting Class Constants**
- More Methods!

New: Class Constants

A fixed value visible (in-scope) to the whole program (the entire ***class***).

Value is set at declaration, **cannot** be reassigned – value is ***constant***.

```
public static final type NAME_OF_CONSTANT = expression;
```

Practice: Think

sli.do #cse121

```
public static final int COUNT = 7;

public static void main(String[] args) {
  int count = 5;
  line(count);
  System.out.println("COUNT is: " + COUNT);
  System.out.println("count is: " + count);
}

public static void line(int count) {
  for (int i = 1; i <= count; i++) {
    System.out.print("*");
  }
  count++;
  System.out.println();
}
```

What will be the **last line of output** from this code?

A. `count is: 1`

B. `count is: 5`

C. `count is: 6`

D. `count is: 7`

Practice: Pair

sli.do #cse121

```
public static final int COUNT = 7;

public static void main(String[] args) {
  int count = 5;
  line(count);
  System.out.println("COUNT is: " + COUNT);
  System.out.println("count is: " + count);
}

public static void line(int count) {
  for (int i = 1; i <= count; i++) {
    System.out.print("*");
  }
  count++;
  System.out.println();
}
```

What will be the **last line of output** from this code?

A. `count is: 1`

B. `count is: 5`

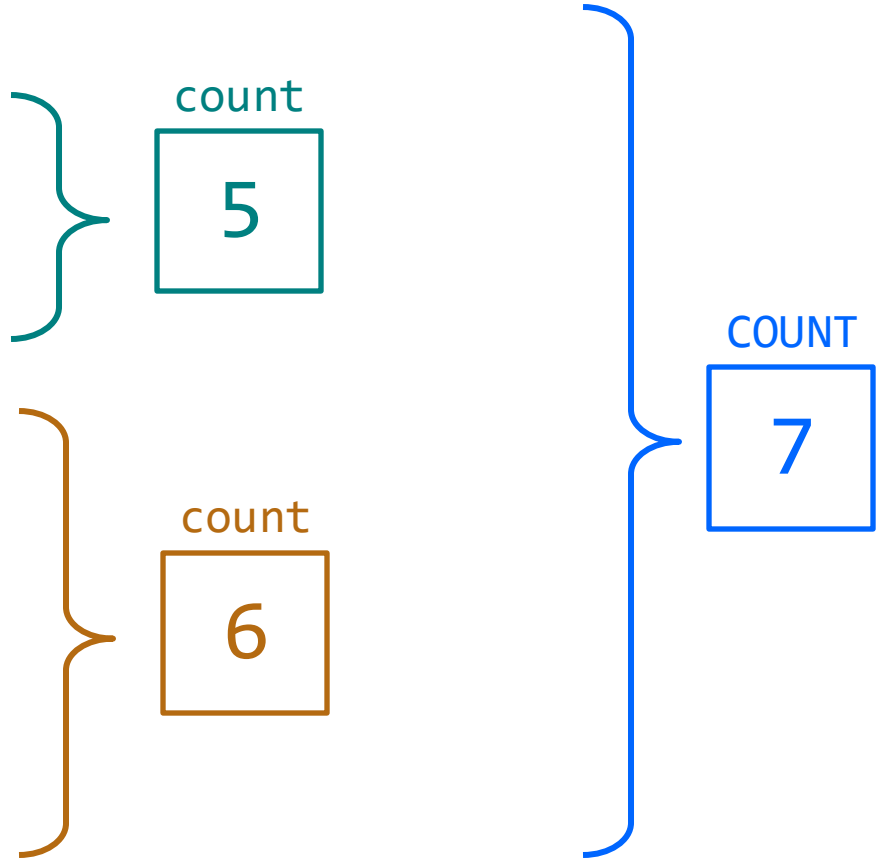
C. `count is: 6`

D. `count is: 7`

Walkthrough: Counting Counts

```
public static final int COUNT = 7;
public static void main(String[] args) {
  int count = 5;
  line(count);
  System.out.println("COUNT is: " + COUNT);
  System.out.println("count is: " + count);
}

public static void line(int count) {
  for (int i = 1; i <= count; i++) {
    System.out.print("*");
  }
  count++;
  System.out.println();
}
```



Up Next: More Methods!

- Announcements, Reminders
- P1 Preview
- Revisiting Class Constants
- **More Methods!**

PCM: Returns

Returns allow us to send values *out of a method*

```
public static <type> myMethod(int num) {
    System.out.print(num + " is the best!");
    ...
    return <value of correct type>
}
```

`return` does 3 things:

1. **Evaluate** the expression
2. **Return** this value to where the method was called
3. **Exit** the method

Calling a method that returns a value…

```
<type> result = myMethod(42);
```

Recall: Math

Method	Description
`Math.abs(`*value*`)`	**Returns** the absolute value of *value*
`Math.ceil(`*value*`)`	**Returns** *value* rounded up
`Math.floor(`*value*`)`	**Returns** *value* rounded down
`Math.max(`*value1*`, `*value2*`)`	**Returns** the larger of the two values
`Math.min(`*value1*`, `*value2*`)`	**Returns** the smaller of the two values
`Math.round(`*value*`)`	**Returns** *value* rounded to the nearest whole number* note: need to cast result to int (it's complicated!)
`Math.sqrt(`*value*`)`	**Returns** the square root of *value*
`Math.pow(`*base*`, `*exp*`)`	**Returns** *base* raised to the *exp* power

Recall: String Methods

Method	Description
`length()`	**Returns** the length of the string.
`charAt(i)`	**Returns** the character at index *i* of the string
`indexOf(s)`	**Returns** the index of the first occurrence of *s* in the string; returns -1 if *s* doesn't appear in the string
`substring(i, j)` or `substring(i)`	**Returns** the characters in this string from *i* (inclusive) to *j* (exclusive); if *j* is omitted, goes until the end of the string
`contains(s)`	**Returns** whether or not the string contains *s*
`equals(s)`	**Returns** whether or not the string is equal to *s* (case-sensitive)
`equalsIgnoreCase(s)`	**Returns** whether or not the string is equal to *s* ignoring case
`toUpperCase()`	**Returns** an uppercase version of the string
`toLowerCase()`	**Returns** a lowercase version of the string